Tuning Systems: From Composition to Performance

JANE HILLSTON

School of Informatics, University of Edinburgh, Scotland UK
Email: Jane.Hillston@ed.ac.uk

This paper gives a summary of some of the work of the PEPA project, work which attracted the 2004 Roger Needham Award from the BCS. Centred on the PEPA modelling formalism, the project has been a happy interaction of theory and application. Theoretical developments have been tested and validated by application to a wide range of problems including the originally intended application domain of performance analysis of computer and communication systems, but also a number of unforeseen novel application areas.

Keywords: Performance modelling, stochastic process algebra, PEPA, compositional modelling, decomposed solution

Received ??; revised 2005; accepted ??

1. INTRODUCTION

The PEPA project started in Edinburgh in 1991. It was motivated by problems encountered when carrying out performance analysis of large computer and communication systems, based on numerical analysis of Markov processes. Performance analysis seeks to predict the behaviour of a system with respect to dynamic properties such as the number of requests that can be satisfied per unit time (throughput) and response time. Markov processes, whilst offering general applicability, are difficult to construct for large systems so an intermediate system description language is often used. In the early nineties the most common of these were *queueing networks* and *Stochastic Petri Nets* (SPN). Queueing networks, whilst very powerful when applicable, have limited expressiveness and lack formal interpretation. SPN models have formal interpretation but do not have the explicit structure found in queueing networks, which greatly eases model construction.

Process algebras such as CCS and CSP [76, 63] offer a compositional description technique supported by apparatus for formal reasoning. From a performance perspective, however, they lack the timing information essential to derive performance estimates: actions are instantaneous and choices are non-deterministic. Performance Evaluation Process Algebra (PEPA) sought to address these problems by the introduction of a suitably quantified process algebra. Associating an exponentially distributed delay with each activity of the process algebra, PEPA is a language suitable for generating a continuous time Markov chain (CTMC).

The PEPA project has sought to investigate and exploit the interplay between process algebra and the underlying continuous time Markov chain (CTMC).

The remainder of this paper is structured as follows. In Section 2 a more detailed account of the motivation and background of the project is given. Some of the theoretical developments of the PEPA project are described in Section 3. It is unwise practice to develop theory without checking it against real applications and some of the case studies which have been undertaken using PEPA are outlined in Section 4. This section also gives an account of the tool support for PEPA and recent efforts to extend the applicability of PEPA modelling. To conclude, some directions for future and on-going work are presented in Section 5.

2. BACKGROUND

In this section we explain in more detail the context in which PEPA was developed, that of performance modelling of computer and communication systems using continuous time Markov chains (CTMC). This is followed by a brief overview of process algebras, and an introduction to the idea of *stochastic* process algebra.

2.1. Performance Modelling using CTMC

There are a variety of approaches available for the performance evaluation of computer and communication systems. If the system exists it may be possible to monitor the system directly. However, in general, such an approach is time-consuming, difficult and lacks generality. Therefore it is often preferrable to model the system rather than use such direct experimentation. Indeed when the system is yet to be constructed, modelling is the only option.

Performance models may be analysed by simulation, numerical solution or analytical solution. Simulation models have the advantages of being insensitive to state space size. Unfortunately such models are time-consuming to analyse and bring the intellectual burden of evaluating the trustworthiness of the results by the calculation of confidence intervals.

In contrast analytic solution, in which an expression for the performance measure of interest is derived in terms of the input parameters of the model, can be extremely efficient to use. However, constructing such solutions is very much the domain of the expert and typically each system requires a bespoke solution.

Numerical solution of a Markov chain offers a compromise between these two extremes. Some assumptions about the system, particularly with respect to the timing of events, are needed. But the resulting models are relatively straightforward to solve, relying only on simple linear algebra techniques. For moderately sized models the generator matrix of the Markov chain can be stored as a dense matrix, admitting direct solution methods with good numerical accuracy. For larger models sparse matrices are needed, necessitating the use of iterative solution techniques with some loss of numerical precision. The largest models require yet more ingenuity in the representation of the matrix using Kronecker or BDD-based storage. Here convergence becomes an issue plus these storage schemes inevitably lead to much longer solution times, because they are data structures tuned for compactness not speed of access.

2.2. Process Algebra

Process algebras are abstract languages used for the specification and design of concurrent systems. The most widely known process algebras are Milner's *Calculus of Communicating Systems* (CCS) [76] and Hoare's *Communicating Sequential Processes* (CSP) [63]. The stochastic process algebras take inspiration from both these formalisms.

In the process algebra approach systems are modelled as collections of entities, called *agents*, which execute atomic *actions*. These actions are the building blocks of the language and they are used to describe sequential behaviours which may run concurrently, and synchronisations or communications between them.

In CCS two agents communicate when one performs an action, a say, while the other performs the complementary action $\bar{a}$. The resulting communication action has the distinguished label τ, which represents an *internal* action that is invisible to the environment. Agents may proceed with their internal actions simultaneously but it is important to note that the semantics given to the language imposes an interleaving on such concurrent behaviour.

The communication mechanism in CSP is different as there is no notion of complementary actions. In CSP two agents communicate by simultaneously executing actions with the same label. Since during the communication the joint action remains visible to the environment, it can be reused by other concurrent processes so that more than two processes can be involved in the communication (*multiway synchronisation*). This is the communication mechanism adopted by most of the SPA languages.

In either case the language is given a small-step structured operational semantics [80], and this is used to generate a *labelled transition system*. This can be regarded as a *derivative tree* or *graph*, in which language terms form the nodes and transitions (actions) are the arcs, which records all the possible evolutions of a language expression or model. This graph can be used for functional verification. For example (see Figure 1):

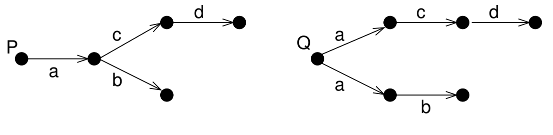
Reachability analysis considers whether there is an evolution of the system which will arrive at a particular state or exhibit a particular behaviour, for example establishing *freedom from deadlock* or *livelock*.

Specification matching contrasts a process algebra expression of the expected behaviour of the system, a *specification*, with a model of the actual implementation. If under an appropriate notion of equivalence the two are equivalent it can be verified that the implementation will deliver the expected behaviour.

Model checking. Many process algebras are equipped with complementary logics which allow properties of the system or its evolution to be expressed[1]. Model checking is a procedure for establishing whether a particular property ϕ will be respected by the system.

Various forms of equivalence relations have been defined for process algebras, based on the structured operational semantics, and they play a fundamental role in model analysis. An important class of relations are based on the notion of *bisimulation*: two agents are bisimilar if, from the perspective of an observer, each is able to mimic the behaviour of the other.

Consider the agents, P and Q, below:



Although the two are trace equivalent, i.e. they can generate identical sequences of actions, they are not bisimulation equivalent because when Q has performed an a action it cannot offer the choice of a b or a c in the manner of P.

2.3. Stochastic Process Algebra

In order to carry out performance analysis of a system, it is essential to record information about the

[1]Specifications can be expressed in such a logic as an alternative to a process algebra model.

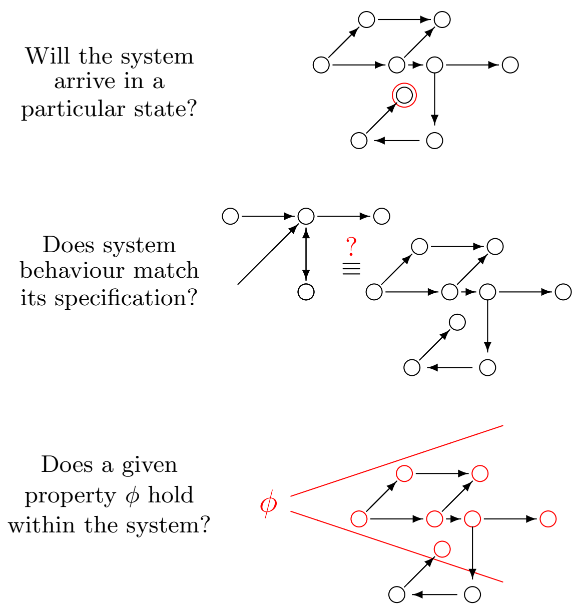

Will the system arrive in a particular state?
Does system behaviour match its specification?
?
≡
Does a given property ϕ hold within the system?
ϕ

FIGURE 1. Functional analysis of process algebra

timing characteristics of the system and the relative probabilities of alternative behaviours. Without this quantified information it is not possible to derive quantitative measures such as expected response time or throughput. Therefore in order to create a process algebra suitable for performance modelling this quantification was added by associating a random variable with all of the actions of the algebra [52]. In the case of PEPA it is assumed that these random variables are governed by a negative exponential distribution [55]. This is because it is only in that case that it is possible, in general, to associate a CTMC with the process algebra model. Explicit probabilities need not be used to differentiate alternative behaviours — implicit probabilities may be derived from the relative timings of the actions involved.

The benefits of adding quantification can be seen if we reconsider the functional analysis scenarios presented in Figure 1. In each case we can enrich the question asked to take into account timing or probability information. For example, we can now query how long it might take to reach a particular state or behaviour, whether the probability profiles of the specification and the implementation models match, and the probability that a particular property holds.

3. THEORY

The theoretical development underpinning PEPA has focused on the interaction between the process algebra and the underlying mathematical structure, the CTMC. The work can be broadly categorised into three areas:

- Designing the language
- Manipulating models
- Solving models and deriving measures

These will be discussed in the following subsections.

3.1. Designing the language

In the early 1990s several stochastic process algebras motivated by the desire to add quantification to process algebra models and make them suitable for performance modelling, appeared in the literature. These included TIPP [47] from the University of Erlangen, EMPA [6] from the University of Bologna and SPADE [83] from Imperial College, in addition to PEPA. PEPA was the first language to be developed with the intention of generating a CTMC which could be solved numerically, but versions of TIPP and EMPA from around the same time were similarly Markovian based.

As previously mentioned, PEPA extends classical process algebra by associating a random variable, representating duration, with every action. These random variables are assumed to be negative exponentially distributed and this leads to a clear relationship between the process algebra model and a CTMC. Via this underlying CTMC performance measures can be extracted from the model.

PEPA models are described as interactions of *components*. Each component can perform a set of actions: an action $a \in \mathcal{A}ct$ is described by a pair (α, r), where $\alpha \in \mathcal{A}$ is the *type* of the action and $r \in \mathbb{R}^+$ is the parameter of a negative exponential distribution governing its duration. Whenever a process P can perform an action, an instance of the probability distribution is sampled: the resulting number specifies how long it will take to complete the action in this instance.

A small but powerful set of combinators is used to build up complex behaviour from simpler behaviour. The combinators are familar from classical process algebra: prefix, choice, parallel composition (*cooperation*) and abstraction (*hiding*). Each of the combinators is informally introduced below in terms of a very simple model of a web-based information system, and the formal operational semantics are shown in Figure 2.

Prefix (.): A component may have purely sequential behaviour, repeatedly undertaking one activity after another and eventually returning to the beginning of its behaviour. A simple example is a web server, which allows one data transfer at a time. Each browser requiring web pages etc. will need to acquire access to the server and only when the transfer is complete will the server be released and available again for acquisition.

$$Server \overset{def}{=} (get, \top).(downld, \mu).(rel, \top).Server$$

In some cases, as here, the rate of an action is outside the control of this component. Such actions are carried out jointly with another component, with this component playing a passive role. For example, the

Prefix

$$\frac{}{(\alpha, r).E \xrightarrow{(\alpha,r)} E}$$

Choice

$$\frac{E \xrightarrow{(\alpha,r)} E'}{E + F \xrightarrow{(\alpha,r)} E'}$$

$$\frac{F \xrightarrow{(\alpha,r)} F'}{E + F \xrightarrow{(\alpha,r)} F'}$$

Cooperation

$$\frac{E \xrightarrow{(\alpha,r)} E'}{E \underset{L}{\bowtie} F \xrightarrow{(\alpha,r)} E' \underset{L}{\bowtie} F} \quad (\alpha \notin L)$$

$$\frac{F \xrightarrow{(\alpha,r)} F'}{E \underset{L}{\bowtie} F \xrightarrow{(\alpha,r)} E \underset{L}{\bowtie} F'} \quad (\alpha \notin L)$$

$$\frac{E \xrightarrow{(\alpha,r_1)} E' \quad F \xrightarrow{(\alpha,r_2)} F'}{E \underset{L}{\bowtie} F \xrightarrow{(\alpha,R)} E' \underset{L}{\bowtie} F'} \quad (\alpha \in L)$$

$$\text{where} \quad R = \frac{r_1}{r_\alpha(E)} \frac{r_2}{r_\alpha(F)} \min(r_\alpha(E), r_\alpha(F))$$

Hiding

$$\frac{E \xrightarrow{(\alpha,r)} E'}{E/L \xrightarrow{(\alpha,r)} E'/L} \quad (\alpha \notin L)$$

$$\frac{E \xrightarrow{(\alpha,r)} E'}{E/L \xrightarrow{(\tau,r)} E'/L} \quad (\alpha \in L)$$

Constant

$$\frac{E \xrightarrow{(\alpha,r)} E'}{A \xrightarrow{(\alpha,r)} E'} \quad (A \overset{\mathit{def}}{=} E)$$

FIGURE 2. PEPA Structured Operational Semantics

server is passive with respect to the *get* action and this is recorded by the distinguished symbol, $\top$ (called "top").

Choice (+): A choice between two possible behaviours is represented as the sum of the possibilities. For example, if we consider a browser in the information system, displaying the current data may have two possible outcomes: demand for access to data available in the local cache (with probability p_1) or demand for access to data stored at the remote server (with probability $p_2 = 1 - p_1$). These alternatives are represented as shown below:

$$\begin{aligned} \mathit{Browser} \overset{\mathit{def}}{=}\ & (\mathit{display}, p_1\lambda).(\mathit{cache}, m).\mathit{Browser}\ + \\ & (\mathit{display}, p_2\lambda).(\mathit{get}, g).(\mathit{downld}, \top).(\mathit{rel}, r).\mathit{Browser} \end{aligned}$$

A *race condition* governs the behaviour of simultaneously enabled actions and the continuous nature of the probability distributions ensures that the actions cannot occur simultaneously. Thus a sum will behave as either one summand or the other. When an action has more than one possible outcome, e.g. the *display* action in the browser, it is represented by a choice of separate actions, one for each possible outcome. The rates of these actions are chosen to reflect their relative probabilities.

Cooperation ($\underset{L}{\bowtie}$): We have already anticipated that the browser and the server in the example will be working together within the same system. This will require them to *cooperate* when the browser needs to download data which is not available locally. In contrast, the local activities of the browser can be carried out independently of the server. Cooperation over given actions is reflected in the cooperation by the *cooperation set*, $L = \{\mathit{get}, \mathit{download}, \mathit{rel}\}$ in this case. Actions in this set require the simultaneous involvement of both components. The resulting action, a *shared* action, will have the same type as the two contributory actions and a rate reflecting the rate of the action in the slowest participating component. This is discussed in more detail in the following subsection.

If, for simplicity, we assume that the system consists of just two browsers, the system is represented as the cooperation of the browsers and the server as follows:

$$\begin{aligned} \mathit{WEB} \overset{\mathit{def}}{=}\ & \big(\mathit{Browser} \parallel \mathit{Browser}\big) \underset{L}{\bowtie} \mathit{Server} \\ & L = \{\mathit{get}, \mathit{download}, \mathit{rel}\} \end{aligned}$$

The combinator $\parallel$ is a degenerate form of the cooperation combinator, formed when two components behave completely independently, without any cooperation between them (i.e. $L = \emptyset$), as in the case of the two browsers.

Abstraction (/): It is often convenient to hide some actions, making them private to the component or components involved. The duration of the actions is unaffected, but their type becomes hidden, appearing instead as the unknown type τ. Components cannot synchronise on τ. For example, as we further develop the model of the information system we may wish to hide the access of a browser to its local cache. This might lead to a new representation of the browser:

$$\mathit{Browser}' \overset{\mathit{def}}{=} \mathit{Browser}/\{\mathit{cache}\}$$

and a corresponding new representation of the system:

$$\begin{aligned} \mathit{WEB}' \overset{\mathit{def}}{=}\ & \big(\mathit{Browser}' \parallel \mathit{Browser}'\big) \underset{L}{\bowtie} \mathit{Server} \\ & L = \{\mathit{get}, \mathit{download}, \mathit{rel}\} \end{aligned}$$

Use of the hiding combinator in this way has two implications. Firstly, it ensures that no components added to the model at a later stage can interact, or interfere, with this action of the browser. Secondly, private actions are deemed to have no contribution to the performance measures being calculated and this might subsequently suggest simplifications to the model.

Throughout the simple example above we have used constants (names) such as *Server* to associate identifiers with behaviours. Using recursive definitions we are able to describe components with infinite behaviours.

3.1.1. Cooperation

Communication or parallel composition is the essence of compositionality in process algebras. It gives structure to models, indicating which actions may be undertaken concurrently, and which cannot. In PEPA there is no concept of conjugate or complementatry actions, as there is CCS, because the aim was to capture something more general than a communication. Thus there need not be strict input and output roles assigned to the participants. Instead a multiway synchronisation framework is adopted as in CSP. This means that components or agents jointly perform actions of the same type, when the parallel composition dictates it. Note that in PEPA the cooperation combinator is in fact a family of combinators, since its meaning varies according to the contents of the cooperation set L.

Additional consideration is needed since the actions which are to be performed jointly may each have been assigned rates (durations) in their respective components. The issue of what it means for two timed actions to synchronise is a vexed one and the various stochastic process algebras have adopted a variety of solutions to this problem. This issue is discussed in [53] and in detail in Bradley's thesis [14].

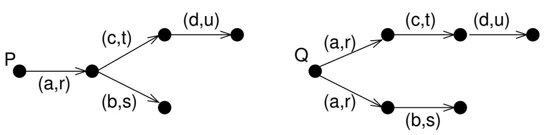
In PEPA it is assumed that each component has *bounded capacity* to carry out activities of any particular type, determined by the *apparent rate*. For a component P and action type α, the apparent rate of α in P, denoted $r_\alpha(P)$, is the sum of the rates of each α action enabled in P. This corresponds to the rate at which P appears to an external observer to carry out an α action, due to the superposition principle of the negative exponential distribution. The definition of cooperation in PEPA is based on the assumption that a component cannot be made to exceed its bounded capacity, meaning that the apparent rate of the shared action will be the minimum of the apparent rates of the components involved.

3.1.2. Semantics and Equivalence relations

The semantic rules of PEPA generate a labelled transition system, just as in the case of classical process algebra. However there are some significant differences introduced by the inclusion of quantified information. In particular it is important to note that the semantics gives rise to a *multi-transition* system i.e. it is not sufficient to record the existence of a transition or arc between two nodes. The multiplicity of the transition is also important. This is because the apparent rate of a term which has two copies of the same arc, generated by two instances of the same action, will differ from that of a term with only one instance.

Once a derivation graph has been generated for a particular model this forms the basis of the underlying CTMC on which performance analysis will be carried out. A state of the CTMC is associated with each node of the graph, and the transition rate between states is simply the sum of the rates of actions labelling arcs between the corresponding nodes. Thus each syntactic term of the PEPA model during its evolution corresponds to a state of the CTMC. It is established in [55] that this generates a unique Markov process.

PEPA has been equipped with a number of equivalence relations which have been shown to be useful for a variety of purposes [55]. The most significant is *strong equivalence*, sometimes termed *Markovian bisimulation*. Just as with the bisimulation for classical process algebra, the central notion here is that each of the pair of components should be able to mimic the behaviour of the other from the perspective of an external observer. This observer is now assumed to have the ability to time the behaviour over many repetitions and thus deduce information about the apparent rates of actions. This means that for components to be strongly equivalent they must have the same apparent rate for all action types. Therefore if we consider again the two processes, P and Q, considered earlier, now enhanced with activity rates,



we can immediately deduce they are not bisimilar because the apparent rates with respect to a in the initial states are not the same. Note that this is a bisimulation in the same style as the bisimulation defined by Larsen and Skou for a probabilistic variant of CCS [73].

Kemeny and Snell, established in 1960 that if we partition the state space of a CTMC, and then form a new stochastic process in which each partition forms a state and the transition rate between states is the superposed transition rate of all transitions in one partition to the other, this stochastic process will satisfy the Markov property, if and only if, the partition has a property called *lumpability* [70]. An important property of strong equivalence in PEPA is that it induces a lumpable partition on the underlying CTMC. This forms the basis of an exact model reduction technique, termed *aggregation* which is discussed in the next subsection.

3.2. Manipulating models

PEPA, like all state-based modelling techniques, suffers from problems of *state space explosion*. The compositionality of the process algebra can greatly aid model construction, but it can readily result in a model which is too large to be solved directly. As explained earlier, research into techniques which can reduce the state space of models, or otherwise make them more amenable to solution, has been an active area of research in performance modelling for over 20 years. Two such techniques are *model simplification* and *model aggregation*, and the application of these techniques in the process algebra setting has been investigated. The challenge for PEPA has been to define such model manipulation techniques in the context of the process algebra, in such a way that they can subsequently be applied automatically, based on the formally defined equivalence relations.

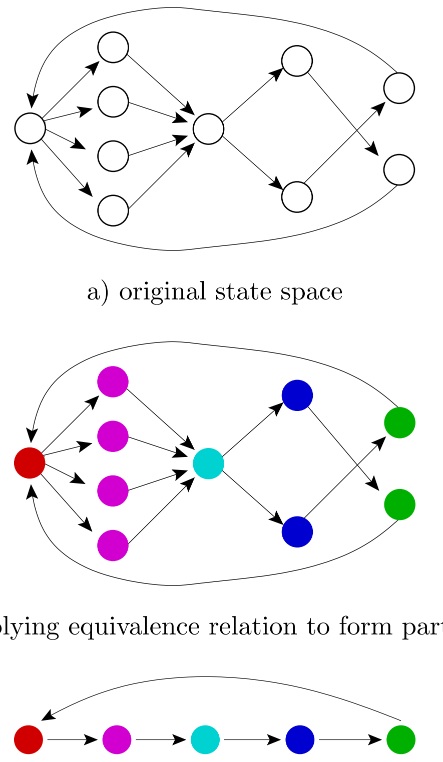
3.2.1. Model simplification

In model simplification the objective is to replace one model by another. Thus an equivalence relation is used to establish behavioural or observational equivalence *between models*. The aim is to find a replacement model which is more desirable from a solution point of view, e.g. smaller state space, special class of model, etc. Once the desirable model has replaced the original, the underlying CTMC is generated as usual, associating one state with each node in the labelled transition system generated by the semantics. Equivalence relations that have been used in this way are *weak isomorphism* [55, 24] and *strong bisimulation* [55] in PEPA.

3.2.2. Model aggregation

In model aggregation the objective is to take a more abstract view of the system, and thus regard the model at a coarser granularity. Here an equivalence relation is used to establish behavioural or observational equivalence *between states within a model*. In effect this results in an alternative mapping from the labelled transition system, given by the semantics of the model, to the underlying CTMC. The equivalence relation is used to partition the nodes of the labelled transition system into equivalence classes. Then, instead of the usual one-to-one correspondence between nodes and states, one state in the CTMC is associated with each equivalence class of nodes (see Figure 3) [54].

Establishing the link between strong equivalence and lumpability in the underlying Markov process forms the basis for an automatic procedure to reduce the size of models. Moreover, since the equivalence relation is a *congruence* the reduction can be applied to isolated components of the model meaning that the state space of the complete model need never be constructed [54]. An algorithm to apply this technique on-the-fly, at the process algebra level, during state space generation has been developed and implemented for PEPA models [42].


a) original state space

b) applying equivalence relation to form partition

c) aggregate, reducing each partition to a single state

FIGURE 3. State-state equivalence for aggregation

3.3. Solving models

Despite the successes that have been gained in techniques for model manipulation, it still remains the case that in many instances the matrix characterising the underlying CTMC, and the corresponding steady state probability vector, are simply too large to be readily stored on standard computing equipment. A variety of approaches to this problem for numerical solution of CTMCs have appeared in the literature, including using disk-based storage [33], Kronecker and BDD-based representation of these entities [57, 51] and decomposed solutions of various forms [56].

Much work on PEPA has studied the use of decomposed solutions. It is clear that there is great advantage to be gained if the compositional structure of a PEPA model can be used during model solution, i.e. if the CTMCs corresponding to the components can be solved separately and their solutions combined to obtain a solution, exact or approximate, of the whole CTMC. This work is discussed in this subsection.

In many cases the techniques which are applied are well-known at the CTMC level. The advantage of characterising the corresponding class of PEPA models is that by "lifting" the definition from the stochastic process level to a formally defined high-level modelling paradigm we can facilitate the automatic detection of these structures when they occur, thus avoiding the construction of the original CTMC.

3.3.1. *Product form models*

One class of CTMCs which are susceptible to an efficient solution technique are those which exhibit a *product form* equilibrium distribution. Consider a CTMC $X(t)$, whose state space $\mathcal{S}$ is of the form $\mathcal{S} \subseteq S_1 \times S_2$, i.e. each state $\mathbf{s} = s_1 \times s_2$ contains two pieces of information capturing different aspects of the current state. In general, these aspects may be dependent in many ways. When the process $X(t)$ exhibits a product form solution, i.e. the steady state probability of an arbitrary state $\mathbf{s}$, $\pi(\mathbf{s})$, can be expressed as $\pi_1(s_1) \times \pi_2(s_2)$, it indicates that these different aspects of the state description are independent with respect to steady state.

Product form distributions have been widely used in the analysis of queueing networks and, due to their efficient solution, have contributed to the popularity of queueing networks in performance analysis. For example, Jackson networks [66] and their generalisation BCMP-networks [3], have been widely employed. In these cases the underlying CTMC is known to have a *reversible* or *quasi-reversible* structure.

Work on finding PEPA models which give rise to product form solutions has drawn on the previous work on queueing networks. Essentially this can be seen as an investigation of when components *interact* and yet remain *statistically independent* at steady state. It is clear that when a PEPA model consists of completely independent components, i.e. $P \parallel Q$, the steady state distribution will have product form:

$$\pi(P \parallel Q) = \pi_P(P) \times \pi_Q(Q)$$

where π_P and π_Q are the steady state distributions over the local states of P and Q respectively. However, few real systems consist of components which are independent in this way. The challenge has been to find circumstances in which components P and Q which interact, still exhibit statistical independence. A number of classes of such models have been identified:

Reversible models Informally, a reversible Markov process is one which behaves identically when we observe it with time reversed as when we observe it with time flowing forward. More formally, an irreducible, stationary CTMC $X(t)$ is *reversible* if it satisfies the detailed balance equations:

$$\pi(j)q(j,k) = \pi(k)q(k,j) \qquad (1)$$

where $q(j,k)$ is the instantaneous transition rate from state j to state k and $\pi(\cdot)$ is the steady state probability distribution.

An initial study of the structure of the state space of SPA models giving rise to reversible CTMCs was presented by Bhabuta *et al.* in [7]. In [61], Hillston and Thomas, identify syntactic conditions which a PEPA model must satisfy in order for the underlying process to be reversible. The problem is tackled in two stages. First, a basic class of sequential components which give rise to reversible structures are identified. Then, assuming that a known class of reversible PEPA components exist, the authors investigate under what circumstances the conditions for reversibility will be preserved if reversible components are composed using the combinators of the PEPA language.

Fundamental to the basic class of reversible sequential components is the notion of a *reverse pair*. A pair of action types $(\alpha, -\alpha)$ form a reverse pair if, in any state, any α transition leads to a state in which a $-\alpha$ transition leads back to the original state. This ability to "undo" any transition in the subsequent transition seems to be fundamental to reversibility. It is clear to see that this is a necessary condition for equation 1 to be satisfied.

Quasi-reversible models Like reversibility, *quasi-reversibility* originates in queueing theory. Formally, a stationary CTMC $X(t)$ is *quasi-reversible* if, for all times t_0 the state $X(t_0)$ is independent of

1. the input process after t_0 and
2. the output process before t_0.

Rather than the detailed balance equations which characterised reversibility, a quasi-reversible process satisfies *partial balance equations*:

$$\pi(i) \sum_{j \in S'} q(i,j) = \sum_{j \in S'} \pi(j)q(j,i) \qquad (2)$$

for all states i and a corresponding subset of states S'. More details of the definition of quasi-reversibility can be found in [69].

In [48], a PEPA characterisation of this class is presented. As in the work on reversibility, the approach is to first find simple instances of PEPA processes which give rise to quasi-reversible structure in their underlying CTMC. Then, conditions are established under which these components can be composed whilst maintaining the quasi-reversible property. Again the notion of a *reverse pair* is important and strong restrictions are placed on the interactions between components: each must be a *flow cooperation*. This means that the "positive" half of a reverse pair in one component is carried out in cooperation with the "negative" half of a reverse pair in another.

Routing process models Sereno's work, reported in [81], derives product form criteria for PEPA models based on earlier work on product form criteria for SPN [50]. The SPN results rely on defining a Markov chain whose states correspond to the transitions of the SPN, the so-called *routing chain*.

This chain exists only when severe restrictions are placed on the forms of synchronisation and resource contention which can be represented in the net.

Sereno's approach for PEPA is completely analogous to the earlier work on SPN—he defines a Markov chain in which the states correspond to the actions of the SPA model. This is called the *routing process*. Sereno shows that if the state space of the routing process can be partitioned into equivalence classes of enabling actions (roughly speaking, one action *enables* another if the post-set of one is the pre-set of the other; we take the transitive closure of that relation), then a product form solution exists. Moreover the partition forms the basis for the decomposition.

Boucherie resource contention models Other classes of models have been considered in which the interaction between components is indirect i.e. the components themselves are composed in parallel (without cooperation) in the model definition, but they compete over cooperation with a third component. Boucherie characterised a class of Markov chains which fit in this framework. In his definition, otherwise independent CTMCs compete for exclusive access to shared resources, causing blocking while a resource is held [11].

In [62] Hillston and Thomas characterise this class of CTMC in PEPA. As in the underlying CTMCs, the PEPA models consist of non-interacting components which give rise to the constituent processes of the underlying CTMC. These components compete, via synchronisation with *resource components*. A PEPA component is termed a *resource* if it is never free to act independently. The general form of these process algebra terms and the resulting product form is, schematically:

$$\pi\left((P \parallel Q) \underset{L}{\bowtie} R\right) = B \times \pi_P(P \underset{L}{\bowtie} R) \times \pi_Q(Q \underset{L}{\bowtie} R)$$

where the component R represents the resource, π_P and π_Q are the steady state distributions over the derivatives of $P \underset{L}{\bowtie} R$ and $Q \underset{L}{\bowtie} R$ respectively, and B is the normalising constant. The decomposition is formed by considering each of the model terms (P and Q in this case) acting in cooperation with the resource (R) in isolation. Although presented here informally, these conditions are defined as formal syntactical conditions which can be checked on the model specification.

Queueing discipline models In his PhD thesis [24], Clark defined a new combinator $Q_{A,\xi}$ for PEPA which forces sequential components within its scope to observe *first-come-first-served* (FCFS) discipline with respect to action types within the set A. Moreover the rates of activities of those types are no longer controlled by the individual components but by the vector ξ. This is a *derived* combinator, meaning that any expression involving the combinator can be re-expressed using the existing PEPA combinators. In particular for a set of components, $S_1, \dots, S_n$,

$$Q_A(S_1, \dots, S_n) \equiv (S_1 \parallel \cdots \parallel S_n) \underset{M_\xi}{\bowtie} R_\xi$$

for suitably chosen M_ξ and R_ξ.

This class of models is shown to be *insensitive* and therefore to have a product form solution so that the steady state probability of the complete model can be written as an expression involving the steady state probabilities of the individual models solved in isolation. In [28] it is established that this class of models is related to BCMP queueing networks, capturing infinite server and FCFS stations *from the user's perspective.*

Unlike the other clases which have been discussed above, characterisation does not necessitate the definition of syntactic rules which may be used to check whether any model instance belongs to the class of not. Instead the use of the derived combinator means that models can be constructed with a guaranteed product form solution.

Quasi-separable models As with reversibility and quasi-reversibility, the notion of quasi-separability is one which has been developed in relation to queueing networks, in particular queueing networks in which breakdowns occur [78]. It is assumed that the CTMC is comprised of a number of components and that there are two pertinent pieces of information for each component. A representation of the whole process can then be formed as a pair of vectors, each vector capturing one piece of information for each component. For a process to be *quasi-separable* it must be possible to analyse the behaviour of a component, say component i, given the ith element of the first vector and all elements of the second, or vice versa. This allows the complete process to be reduced to a number of sub-models, each of which contains all the information about exactly one component.

For such processes it is not possible to find the exact solution of the steady state distribution as a product of the local steady state distributions. Nevertheless decomposed solution can lead to exact results for the local steady state distributions and many performance measures.

In [84], Thomas and Gilmore present a characterisation of PEPA models which are quasi-separable. It is assumed in this characterisation that the information which must be included in each decomposed submodel is not distributed between the components but maintained by a single *scheduler* component. There are several conditions on the way in which this component may interact with the other components of the model. Furthermore the individual components have no direct interaction between them—they must be in parallel composition with no synchronisation, i.e. each of these components interacts only with the scheduler. The model is decomposed into a set of models, each comprising of a single component considered with the scheduler, in isolation.

Recent work on product form PEPA models has taken a slightly different form. In Harrison's work on the *Reversed Compound Agent Theorem* (RCAT) the process algebra has been used to establish a framework in which the relationships between different classes of product form CTMCs can be compared [49]. Within this framework Harrison has been able to demonstrate that the product forms which arise in Jackson networks [66] and G-networks [36] are based on the same fundamental mechanisms: this becomes apparent when they are represented in PEPA.

3.3.2. Aggregated decomposed solutions

Product form models have the benefit of yielding the *exact* solution of the complete model. There are a variety of other techniques which have been developed for decomposed solution of CTMCs which impose less stringent conditions on the candidate models but which yield only approximate results. In many cases these are forms of *aggregated decomposed* solutions. Due to the richer interactions between submodels it is not sufficient to only consider the submodels in isolation when forming a solution to the whole model. In addition the decomposed solution involves a stochastic representation of the interactions between the submodels, the *aggregated model*. Some work has been done on considering PEPA, and other SPA, models in this framework.

Time scale decomposition The work on time scale decomposition in SPA is based on the notion of *near completely decomposable* CTMCs [31], and inspired by previous work on time scale decomposition of SPN models [8]. A characterisation of a near completely decomposable CTMC at the matrix level is that the matrix is block structured with elements in the diagonal blocks being at least an order of magnitude larger than elements in the off-diagonal blocks. This implies that the model is made up of subsystems whose internal interactions are much more frequent than the interactions between subsystems. As a consequence it can be assumed that the subsystems reach an internal equilibrium between external interactions.

The initial classification of SPA models susceptible to time scale decomposition [60], relied on a classification of the sequential components within a model into *fast* or *slow*; this in turn was based on a classification of all actions relative to some threshold rate. A component is considered to be fast if it enables fast or passive actions; a component is considered to be slow if it enables only slow actions. Only models comprised of fast and slow components were considered. Submodels were formed by allowing evolution only via fast actions. The aggregated model was formed by allowing evolution between such subsets of states via slow actions [60].

Later work by Mertsiotakis [74], tackles the problem of *hybrid* components—sequential components which cannot be classified as either fast or slow since they enable both fast and slow actions.

Related work on other SPAs includes

- Mertsiotakis and Silva's work on decomposition of a class of SPA models, termed *decision free processes* [75, 74], based on earlier work on throughput approximation in a class of SPN called *marked graphs* [68].
- Bohnenkamp and Haverkort's work on *near-independence* [9], exploiting the notion of near-independence introduced by Ciardo and Trivedi in [23] in the context of SPN.
- This was later expanded by the same authors in [10], which aims to reformulate the underlying Markov process of an SPA model as a set of semi-Markov processes. These are then solved via their embedded Markov chains and evaluations of the time distributions between synchronisations between the SPA components.

4. APPLICATIONS

Developing models of real applications has always been part of the PEPA project. This allows us to demonstrate to ourselves and others that the theory we have developed is useful. It is also a valuable source of inspiration for new theory and future directions.

In this section we give a brief overview of some of the case studies which have been undertaken by ourselves and others using PEPA, give an account of some of the tools supporting modelling with PEPA and then more detailed accounts of two recent projects which seek to make PEPA modelling more accessible to practitioners.

4.1. Case studies

As originally intended, PEPA has been applied to study the performance characteristics of a number of computer and communication systems. Initial examples focussed on well-known standard performance evaluation abstractions such as *multi-server multi-queue* systems [55] and various queueing systems [85]. However over time more realistic case studies emerged, both from the PEPA group and from others. For example, in [27] the performance impact of fault-tolerant protocols within a distributed system framework is evaluated. In [13] Bowman *et al.* develop a model of multimedia traffic characteristics and use it to derive quality of service measures such as jitter, throughput and latency. In an investigation of ways in which to ease the development of parallel database systems, the STEADY group at the Heriot-Watt University proposed the use of *performance estimators*. PEPA was used to verify the output of the performance estimators for a number of particular configurations and therefore improve confidence in the approach [34].

In recent work a group at the PRiSM Laboratory of the University of Versailles are working on a novel active rule-based approach to active networks (networks in which intermediate nodes supplement routing of data with some computation) [12]. A PEPA model was used to study the impact of the "*active*" traffic on the non-active cross-traffic in terms of loss rate and latency within an active switch [58]. Furthermore the models were validated against simulation models of the same system and showed very good agreement [59].

In addition, the formalism has been applied to a number of other problems which are beyond the usual arena of computer performance evaluation.

Inland shipping Luk Knapen of Hasselt applied PEPA to study traffic flow within the inland shipping network of Belgium focussing in particular on the locks and movable bridges.

Robotic workcells Robert Holton of the University of Bradford used PEPA models to analyse the performance and functional correctness of a robotic workcell designed for a automated manufacturing system [39, 64].

Cellular telephone networks A team from the PRiSM Laboratory at the University of Versailles considered a problem of dimensioning in a cellular telephone network. They used a PEPA model to study the impact on call blocking and dropping of allocating bandwidth resources between micro and macro-cell level [35]. They took advantage of automatic aggregation [42].

Automotive diagnostic expert systems Console *et al.* of the University of Turin constructed a PEPA model of an automatic diagnostic system to be deployed in a car. A large number of sensors were placed around the car and some number could trigger an alarm. The role of the PEPA model was to provide probabilistic reasoning to resolve the likely cause of the alarm based on previous observations of the timing and frequency of individual faults [30].

4.2. Tool Support

Case studies of the size and complexity described in the previous subsection are only possible if the modelling process has adequate support. The PEPA formalism has been fortunate to be supported in a number of different tools offering a variety of different analysis techniques. Such support has been a strong factor in encouraging others to use the formalism.

The major tools which support PEPA are summarised in Figure 4 and briefly described in the subsections below.

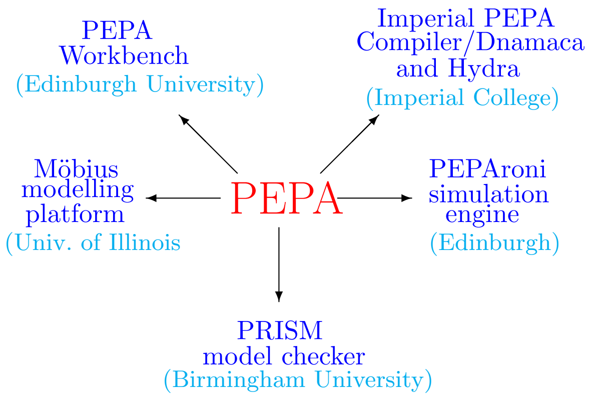


FIGURE 4. Tool support for PEPA modelling

4.2.1. The PEPA Workbench

Initially developed in 1993, the PEPA Workbench has undergone several revisions, but has nevertheless maintained the same core functionality. It provides a parser which can apply the operational sematics to derive the derivation graph capturing all the possible evolutions of the model. It can render this as the infinitesimal generator matrix of a CTMC in formats suitable for both internal numerical solvers (biconjugate gradient algorithm or successive over-relaxation) or external numerical computing platforms such as Maple and Matlab. In addition it includes facilities to automatically derive some performance measures such as throughput, and a one-step debugger which can show the evolution of a model one activity at a time [38, 26].

The tool exists in two versions: the original version written in the functional programming language Standard ML, and a later edition written in Java [65]. The simple high-level tool was used to gain experience and insights. This was subsequently extended to a better engineered, more portable version. The ML edition of the PEPA Workbench is still used in this way, as a testbed for extension of the PEPA language such as PEPA nets [43] and for new algorithms [42]. In contrast, some less research-oriented extensions, such as the inclusion of transient solution facilities [86] are found only in the Java edition.

Generating the CTMC underlying a PEPA model, and finding its steady state probability vector is rarely, if ever, the final objective of PEPA modelling. Formal tool support for querying performance models is an area which has received little attention until recently, despite its practical importance. Whilst some effort has been applied in this direction for PEPA models, it remains an area in which there is much scope for future work.

At the most basic level the modeller wishes to construct a *reward structure* over the state space of the CTMC, to be used in conjunction with the steady state probability vector to derive performance measures. For steady state measures the reward structure is a vector recording a "reward" for each state, although for many states the reward value will be zero. Thus the problem becomes one of identifying the appropriate set of states to attach a non-zero reward to. Clearly, when the CTMC arises from a stochastic process algebra model we prefer to characterise the state at the process algebra level. PEPA analysis tools have been developed which tackle this problem in two distinct ways.

The PEPA State Finder The PEPA State Finder is used with the ML edition of the PEPA Workbench. It identifies subsets of states using regular expression pattern matching, applied to the table of PEPA expressions which make up the state of the model. Recall that there is a one-to-one correspondence between the syntactic forms of the PEPA process as it evolves and the states of the CTMC. The Workbench maintains a table recording this correspondence, and using regular expression pattern matching the PEPA State Finder is able to extract the states of interest. For example, it is possible to use an expression such `*|(next,r).*` to return the state numbers of all the states in which the second component enables a (*next*, *r*) activity. This could then be used to construct a reward structure suitable for calculating the throughput of *next* in the second component: the value r is placed in the reward vector at each position corresponding to a (numerical) state found by the PEPA State Finder.

PML$_\mu$ A more sophisticated means of specifying rewards is described in Clark's PhD thesis [24], and developed around the stochastic logic PML$_\mu$. Inspired by the probabilistic model logic of Laren and Skou, PML [73], PML$_\mu$ is able to differentiate PEPA terms which perform the same activities but at different rates. The key to this is a modification to Hennessy-Milner logic in which the diamond operator becomes decorated with a rate. The semantics of an expression in the logic is a subset of states, and thus logical expressions may be used, in conjunction with a value, to specify a reward structure. Clark extended the ML edition of the PEPA Workbench to include support for PML$_\mu$ and associated reward structures [24].

4.2.2. The Imperial PEPA Compiler

The recently developed Imperial PEPA Compiler (IPC) incorporates an alternative parser for PEPA models [15], thus providing a bridge to alternative analysis tools developed at Imperial College by Knottenbelt and his group [71, 16].

The IPC tool translates an input PEPA model into the Petri net notation provided by Dnamaca [71]. Its support for the PEPA language is comprehensive. Apparent rates are supported, as are anonymous components. These are two advantages over the PEPA-to-PRISM compiler, and a richer class of PEPA models can be analysed by IPC/Dnamaca as a result.

The steady state probability distribution represents the behaviour of the system at equilibrium, where the influence of the initial state of the system is no longer measurable. Some performance measures of interest cannot be derived from the results of steady state analysis. Examples of performance measures in the class of non-equilibrium measurements include *mean time to failure* analysis, as computed in the evaluation of dependable systems. Other examples include the probabilistic quality-of-service guarantees which underpin most commercial service level agreements (SLAs): e.g. the probability that a 10-node cluster should be able to process 3000 database transactions in less than 6 seconds should be greater than 0.915; or a train service should not run more than 10 minutes late more than 20% of the time.

More generally, such measures necessitate the computation of *passage-time quantiles* which detail the probability of passing through the system evolution from a start state to an end state (or set of starting states to a set of end states). The computation of such measures depends on the aggregate time behaviour across a whole system of complex interactions. The computation of passage-time quantiles depends on *transient analysis* of the CTMC, which is more expensive than steady-state analysis in both run-time and memory consumption.

Via IPC, the unique solution capabilities of Dnamaca become available and because of this it is possible to efficiently perform passage time analysis over PEPA models [15, 16]. Start and end points are specified using the concept of *stochastic probes* developed by Argent-Katwala, Bradley and Dingle [2]. Stochastic probes are themselves PEPA components which have been generated from regular expression-based inputs.

4.2.3. PEPAroni simulation engine

Simulation has proved to be a useful alternative to numerical analysis of the underlying CTMC in two cases. Firstly, if the size of the model is prohibitively large for numerical analysis simulation can be used, although issues of run length can arise. Secondly, in the context of extending the expressiveness of PEPA with general distributions [24] numerical solution is no longer exact in most cases. Principally motivated by this second case, Clark implemented a simulator for PEPA models, called the PEPAroni simulator.

4.2.4. The PRISM model checker

PRISM is a probabilistic model checker developed by Kwiatkowska's group at the University of Birmingham. It supports discrete time Markov chains and Markov decision processes as well as CTMCs. The standard input to PRISM is a model described in a simple reactive modules language. PEPA was integrated into the tool via a compiler which translates PEPA models into this language. There was also some work required to extend the model checker to support PEPA's combinators (cooperation and hiding).

Integration into PRISM enables model checking of the CTMC underlying a PEPA model against properties expressed in Continuous Stochastic Logic (CSL) [72]. It also provides access to the efficient numerical solutions of PRISM based on MTBDDs [51] and sparse matrix representation. PRISM has been applied successfully to a number of PEPA (and PEPA net) case studies [45, 41].

4.2.5. The Möbius modelling platform

The Möbius modelling framework [25] was developed at the University of Illinois Urbana-Champaign. It is both a multi-formalism and multi-paradigm modelling tool, i.e. it aims to offer the user a choice of model description techniques and solution methods. Moreover it is designed to allow a model to be composed of submodels which may be expressed in different formalisms. It has a broad spectrum of users in North America. Integrating PEPA into Möbius offered opportunities to present stochastic process algebra to users who were previously unfamiliar with the formalism, and to expore the possibilities of interaction between modelling formalisms [29].

4.3. New application areas

In this section we describe two recent research projects which have sought, in different ways, to extend the applicability of PEPA. In both cases the use of PEPA becomes much more transparent to the users, who interact with the tools through other high-level system descriptions. Whilst the take-up of PEPA has been more widespread than perhaps anticipated it remains the case that the majority of users are academic. Our objective is not to increase the user community of PEPA *per se* but to encourage the use of sound performance analysis as the basis for design and deployment decisions. Thus in these latest developments the use of PEPA is in some ways hidden from the user.

4.3.1. The DEGAS project

In the CEC-funded DEGAS project[2] we have been investigating ways to make performance modelling using PEPA more accessible to software designers who may be unfamiliar with process algebra. We have sought to take advantage of the popularity of the Unified Modelling Language (UML) by providing a way to derive performance measures from a suitably annotated UML model. A schematic view of this process is shown in Figure 5.

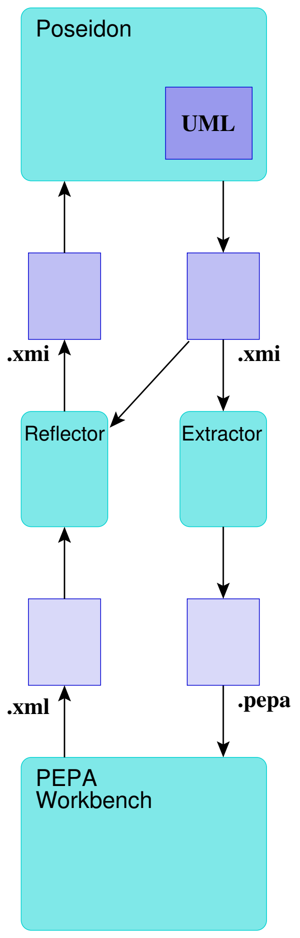


FIGURE 5. Architecture of the DEGAS analysis environment

The key functionality is provided by a pair of software modules, the *extractor* and the *reflector* [20]. These form a bridge between the UML modelling environment and the PEPA tools. Annotations are added to the UML model according to a pre-determined stereotype. The UML is then saved in the usual way in `.xmi` format. The extractor produces a corresponding `.pepa` format which can be loaded into the PEPA Workbench. This allows a steady state probability distribution corresponding to the states of the PEPA model to be derived. However this is still inaccessible to the UML modeller — it is essential that results are reported in terms which make sense to the software designer, i.e. in terms of the original UML model. This functionality is provided by the reflector module which aggregates the steady state probability distribution data to produce suitable annotations to the UML model. This has required a modification to the PEPA Workbench so that results can be written out in XML format. The reflector then combines these with the original `.xmi` file.

[2] *Design Environments for Global ApplicationS* project IST-2001-32072 funded by the FET Proactive Initiative on Global Computing

This scheme is not specific to UML models. For example, an extractor has recently been developed for models developed in BPEL, a web service composition language [77].

4.3.2. The ENHANCE project

The EPSRC-funded ENHANCE project[3] seeks to assist in the development of efficient executions of programs on computational grid environments. The thrust of the project is two-fold

- The use of high-level programming schemes such as Cole's skeletons facilitate correct high-level implementation of systems with complex coordination patterns. These patterns (the "skeletons") are abstracted into a reusable, efficient library of coordination combinators. The use of this library elevates the programming model and eliminates low-level parallel programming errors such as *sends* with unmatched *receives*.
- Predefined performance models (PEPA components) corresponding to these skeletons, as well as the grid infrastructure, can be appropriately parameterised to represent any given configuration. Thus alternative configurations can be evaluated prior to scheduling (and re-scheduling) in order to select the schedule with the best predicted performance.

A prototype tool, AMoGeT, supporting this framework has been developed [5]. This uses the Network Weather Service to query the available grid in order to obtain a short-term forecast of compute load on these machines. It assimilates this information with a PEPA model composed of the present program and grid infrastructure. In fact a set of PEPA models are constructed, one for each candidate schedule. Solving these models within AMoGeT derives performance information which can be used to place processes on processors in order to reduce the makespan of the application overall.

5. FUTURE WORK

As in the past the plan is to continue the development of both new theory and new applications of PEPA in tandem. In this section we give brief overviews of the current work in both areas.

5.1. New theory

PEPA nets Over the last decade mobility has had a major impact on the way we design, implement and manage many computer systems. Mobility may be manifest in the form of devices which change location and spontaneously connect/disconnect, or in the form of executable code which is moved around the network for a variety of reasons. In either case the effect is that the context in which computation is taking place is dynamically changing, and these changes will have consequences for the performance of the system. The PEPA nets formalism has been designed to capture information about mobility and so allow performance models of such systems to be readily and naturally developed [44].

A PEPA net is a stochastic Petri net with coloured tokens. The tokens represent *mobile* objects with state and behaviour, where we use the term mobile loosely to characterise objects which may find themselves in different contexts during execution. The tokens are described using PEPA.

The use of stochastic Petri nets for performance models is well-established [1] and coloured variants, e.g. Stochastic Well-Formed Nets (SWN) [21], have also been developed. However the use of colours in PEPA nets offers something quite distinct — the possibility of differentiating between two types of change of state within a system. Unlike SWN where tokens remain indistinguishable within their colour classes, tokens within PEPA nets are autonomous components. Firings of the net will typically be used to model macro-step (or *global*) changes of state, whereas transitions within the PEPA tokens are typically used to model micro-step (or *local*) changes of state as components undertake activities. Thus modelling with PEPA nets uses both Petri nets and process algebras together as a single, structured performance modelling formalism. Moreover, we have demonstrated that PEPA nets offer some expressivity which is not directly offered by either PEPA or Petri nets [44]. Their modelling capabilities have been shown on a number of case studies including MobileIP[40], a decentralised peer-to-peer emergency medical application [37] and a distributed multi-user role-playing game [46].

New mathematical models As detailed earlier, a major challenge for all discrete state-based performance modelling formalisms (i.e. queueing networks, stochastic Petri nets and stochastic process algebra) is the problem of *state space explosion*. Much research effort of the performance analysis community for the last twenty years has been devoted to tackling this problem, using techniques such as those discussed in Section 3.3. However, recent work in the PEPA group has been studying an alternative approach. In this approach we move away from the assumption of discrete state space and consider instead continuous approximations. Where there are sufficient numbers of components, early indications are that this is a viable alternative to steady state Markovian analysis [4] for some problems.

When the state variables are assumed to be continuous, and activity rates are taken to be constant, the evolution of the system can be described by a set of ordinary differential equations. When

[3]*Enhancing the Performance Predictability of Grid Applications with Patterns and Process Algebras*, EPSRC grant number GR/S21717/01

this is generalised to allow activity rates to be governed by probability distributions rather than being deterministic the evolution of the system can be described by a set of random differential equations [82]. A further generalisation, introducing more uncertainty, is offered by *stochastic differential equations*[79]. The investigation of the use of these alternative mathematical models is the subject of on-going research.

5.2. New application domains

As already mentioned, the elegance of the PEPA modelling language, the available tool support and the access to quantified analysis have attracted researchers from other fields to develop PEPA models. This has lead us to develop models in some novel areas. We conclude by outlining two of these areas below.

Biochemical signalling pathways We have begun exploratory work in the area of biochemical signalling pathways taking the ERK signalling pathway as an example [22]. A PEPA model has been developed to represent the pathway and subsequently analysed using both standard Markovian analysis [18] and the continuous state space approximation technique described above [19].

In the longer term it is anticipated that new description lanugages and solution techniques will be needed for this new domain of application. For example:

- Process algebras model a system by focusing on the *activities* and *agents* who undertake them. Whilst this view of a system comprised of individuals who interact to produce a collective behaviour has strong resonances within the biological community, the forms of activity and interaction which may be witnessed are more complex. For example, the interaction which captures inhibition without blocking is not readily represented in existing process algebras.
- Issues of scalability of analysis mechanisms have been a major concern within the work on performance evaluation, but the systems considered in the biological domain take this problem even further. In many instances existing techniques are not going to adequately cope, suggesting that new analysis techniques, or at least new mappings between the process algebra and analysis techniques, are needed.

Security and timing attacks Many security analyses can be undertaken without information about the timing of message exchanges and therefore the system description does not need to record timing information. However timing attacks are a form of security breach in which the *timing*, rather than just the *ordering*, of events is crucial.

Timing attacks can be mounted by timing the exchange of messages during a session between two parties. The parts of the communication which do not require user interaction have sufficient repeatability that timings of these interactions can be used to derive information. (Any user interaction usually introduces sufficient random delay to mask any information which could have been obtained from the timing information.) Where a secure session between two parties can be monitored by an eavesdropper then information about the time parts of the interaction can be recorded. In the case where there is a difference in the time taken by different types of interactions (e.g. between a successful interactions and one that fails) then information can be leaked from the communication via the time recorded. One approach to this problem is to introduce delays into the faster interactions in order to mask the difference between fast and slow interactions.

In [17] we have investigated the use of PEPA to analyse modified protocols to ensure that the introduced delays do indeed have the desired effect of masking the timing differences between interactions.

ACKNOWLEDGEMENTS

Many people have contributed to the PEPA project in a variety of ways. In particular the author would like to thank Ashok Argent-Katwala, Anne Benoit, Jeremy Bradley, Linda Brodo, Muffy Calder, Catherine Canevet, Graham Clark, Nick Dingle, Amani El-Rayes, Stephen Gilmore, Zully Grant-Duff, Valentin Haenel, Peter Harrison, Robert Holton, Jon Hunter, Leïla Kloul, Marta Kwiatkowska, Vassilis Mertsiotakis, Amdjed Mokhtari, Gethin Norman, Dave Parker, Corrado Priami, Matthew Prowse, Marina Ribaudo, Matteo Sereno, Fotis Stathopoulos, Joanna Tomasik, Nigel Thomas, Feng Wan.

Tool Availability The PEPA tools are mostly available in open-source form under the GNU Public License. The PEPA Workbench and related tools are available from the PEPA Web site at `www.dcs.ed.ac.uk/pepa`.

The Möbius multi-paradigm modelling framework is available from the University of Illinois at `www.crhc.uiuc.edu/PERFORM/mobius-software.html`.

PRISM is available from `www.cs.bham.ac.uk/~dxp/prism/`.

The Imperial PEPA Compiler is available from `www.doc.ic.ac.uk/ipc/`.

REFERENCES

[1] M. Ajmone Marsan, G. Balbo, G. Conte, S. Donatelli, and G. Franceschinis. *Modelling with Generalized Stochastic Petri Nets.* John Wiley, 1995.

[2] A. Argent-Katwala, J.T. Bradley, and N.J. Dingle. Expressing performance requirements using regular expressions to specify stochastic probes over process algebra models. In *Proc. of 4th Int. Workshop on*

Software and Performance, pages 49–58, Redwood Shores, California, USA, January 2004. ACM Press.

[3] F. Baskett, K.M. Chandy, R.R. Muntz, and F.G. Palacios. Open, Closed and Mixed Networks of Queues with Different Classes of Customers. *Journal of the ACM*, 22(2):248–260, April 1975.

[4] A. Benoit, M. Cole, S. Gilmore, and J. Hillston. Enhancing the effective utilisation of grid clusters by exploiting on-line performability analysis. In *Proceedings of the 2nd Grid Performability Workshop*, 2005. to appear.

[5] A. Benoit, M. Cole, S. Gilmore, and J. Hillston. Realistic performance evaluation of skeleton-based grid applications using the Network Weather Service, 2005. To appear in the Computer Journal.

[6] M. Bernardo and R. Gorrieri. A tutorial on EMPA: a theory of concurrent processes with nondet erminism, priorities, probabilities and time. *TCS*, 202:1–54, 1998.

[7] M. Bhabuta, P.G. Harrison, and K. Kanani. Detecting reversibility in Markovian Process Algebra. In *Performance Engineering of Computer and Telecommunications Systems*, Liverpool John Moores University, September 1995. Springer-Verlag.

[8] A. Blakemore and S. Tripathi. Automated Time Scale Decomposition of SPNs. In *Proc. of 5th International Workshop on Petri Nets and Performance Models (PNPM '93)*, Toulouse, 1993.

[9] H. Bohnenkamp and B. Haverkort. Decomposition Methods for the Solution of Stochastic Process Algebra Models: a Proposal. In E. Brinksma and A. Nymeyer, editors, *Proc. of 5th Process Algebra and Performance Modelling Workshop*, 1997.

[10] H. Bohnenkamp and B. Haverkort. Semi-Numerical Solution of Stochastic Process Algebra Models. In C. Priami, editor, *Proc. of 6th Process Algebra and Performance Modelling Workshop*, 1998.

[11] R.J. Boucherie. A Characterisation of Independence for Competing Markov Chains with Applications to Stochastic Petri Nets. *IEEE Trans. on Software Engineering*, 20(7):536–544, July 1994.

[12] M. Bouzeghoub, L. Kloul, and A. Mokhtari. A new active network framework based on active rules. Technical Report 2002/21, PRiSM, Université de Versailles, 2002.

[13] H. Bowman, J. Bryans, and J. Derrick. Analysis of a multimedia stream using stochastic process algebra. In C. Priami, editor, *6th Int. Workshop on Process Algebras and Performance Modelling*, pages 51–69, Nice, September 1998.

[14] J. Bradley. *Towards Reliable Modelling with Stochastic Process Algebras.* PhD thesis, Department of Computer Science, University of Bristol, 1999.

[15] J.T. Bradley, N.J. Dingle, S.T. Gilmore, and W.J. Knottenbelt. Derivation of passage-time densities in PEPA models using IPC: The Imperial PEPA Compiler. In *Proc. of 11th IEEE/ACM Intl. Symposium on Modeling, Analysis and Simulation of Computer and Telecommunications Systems*, pages 344–351. IEEE Computer Society Press, October 2003.

[16] J.T. Bradley, N.J. Dingle, S.T. Gilmore, and W.J. Knottenbelt. Extracting passage times from PEPA models with the HYDRA tool: A case study. In Jarvis [67], pages 79–90.

[17] M. Buchholtz, S. Gilmore, J. Hillston, and F. Nielson. Securing statically-verified communications protocols against timing attacks. In *Proc. of 1st Int. Workshop on Practical Applications of Stochastic Modelling*, pages 61–79, September 2004. To appear in ENTCS.

[18] M. Calder, S. Gilmore, and J. Hillston. Modelling the influence of RKIP on the ERK signaling pathw ay using the stochastic process algebra PEPA. In *Proceedings of BioConcur'04*, London, England, August 2004.

[19] M. Calder, S. Gilmore, and J. Hillston. Automatically deriving ODEs from process algebra models of signalling pathways. To appear in CMSB'05, 2005.

[20] C. Canevet, S. Gilmore, J. Hillston, M. Prowse, and P. Stevens. Performance modelling with UML and stochastic process algebras. *IEE Proceedings: Computers and Digital Techniques*, 150(2):107–120, March 2003.

[21] G. Chiola, C. Dutheillet, G. Franceschinis, and S. Haddad. Stochastic well-formed colored nets and symmetric modeling applications. *IEEE Transactions on Computers*, 42(11):1343–1360, 1993.

[22] K.-H. Cho, S.-Y. Shin, H.-W. Kim, O. Wolkenhauer, B. McFerran, and W. Kolch. Mathematical modeling of the influence of RKIP on the ERK signaling pathway. In C. Priami, editor, *Computational Methods in Systems Biology (CSMB'03)*, volume 2602 of *LNCS*, pages 127–141. Springer-Verlag, 2003.

[23] G. Ciardo and K.S. Trivedi. A Decomposition Approach for Stochastic Petri Net Models. *Performance Evaluation*, 1992.

[24] G. Clark. *Techniques for the Construction and Analysis of Algebraic Performance Models.* PhD thesis, The University of Edinburgh, 2000.

[25] G. Clark, T. Courtney, D. Daly, D. Deavours, S. Derisavi, J. M. Doyle, W. H. Sanders, and P. Webster. The Möbius modeling tool. In *Proc. of 9th Int. Workshop on Petri Nets and Performance Models*, pages 241–250, Aachen, Germany, September 2001.

[26] G. Clark, S. Gilmore, and J. Hillston. The PEPA performance modelling tools. In J. Hillston, editor, *Proc. of 7th Workshop on Process Algebra and Performance Modelling*, Zaragosa, Spain, September 1999. University of Zaragosa Press.

[27] G. Clark, S. Gilmore, J. Hillston, and M. Ribaudo. Exploiting modal logic to express performance measures. In *Computer Performance Evaluation: Modelling Techniques and Tools, Proc. of 11th Int. Conf.*, number 1786 in LNCS, pages 211–227, Schaumburg, Illinois, USA, March 2000. Springer-Verlag.

[28] G. Clark and J. Hillston. Product form solution for an insensitive stochastic process algebra structure. *Performance Evaluation*, 50(2–3):129–151, 2002.

[29] G. Clark and W.H. Sanders. Implementing a stochastic process algebra within the Möbius modeling framework. In de Alfaro and Gilmore [32], pages 200–215.

[30] L. Console, C. Picardi, and M. Ribaudo. Diagnosis and Diagnosability Analysis using Process Algebras. In *Proc. of 11th Int. Workshop on Principles of Diagnosis (DX00)*, Morelia, Mexico, June 2000.

[31] P.J. Courtois. *Decomposability: Queueing and Computer System Applications.* ACM Series. Academic Press, New York, 1977.

[32] L. de Alfaro and S. Gilmore, editors. *Proceedings of the first joint PAPM-PROBMIV Workshop*, volume 2165 of *LNCS*, Aachen, Germany, September 2001. Springer-Verlag.

[33] D.D. Deavours and W.H. Sanders. An efficient disk-based tool for solving large Markov models. *Performance Evaluation*, 33:67–84, 1998.

[34] E. W. Dempster, N. T. Tomov, J. Lü, C. S. Pua, M. H. Williams, A. Burger, H. Taylor, and P. Broughton. Verifying a performance estimator for parallel DBMSs. In *Proceedings of EuroPar (EuroPar'98)*, September 1998.

[35] J.M. Forneau, L. Kloul, and F. Valois. Performance modelling of hierarchical cellular networks using PEPA. *Performance Evaluation*, 50(2–3):83–99, 2002.

[36] E. Gelenbe. Queueing networks with negative and positive customers. *Journal of Applied Probability*, 28:656–663, 1991.

[37] S. Gilmore, V. Haenel, J. Hillston, and L. Kloul. PEPA nets in practice: Modelling a decentralised peer-to-peer emergency medial application. In M. Núñez *et al*, editor, *Applying Formal Methods: Testing, Performance, and M/E-Commerce (EPEW 2004)*, volume 3236 of *LNCS*, pages 262–277. Springer-Verlag, October 2004.

[38] S. Gilmore and J. Hillston. The PEPA Workbench: A Tool to Support a Process Algebra-based Approach to Performance Modelling. In *Proc. of 7th Intl. Conf. on Modelling Techniques and Tools for Computer Performance Evaluation*, number 794 in LNCS, pages 353–368, Vienna, May 1994. Springer-Verlag.

[39] S. Gilmore, J. Hillston, D.R.W. Holton, and M. Rettelbach. Specifications in Stochastic Process Algebra for a Robot Control Problem. *International Journal of Production Research*, 34(4):1065–1080, 1996.

[40] S. Gilmore, J. Hillston, and L. Kloul. PEPA nets. In M.C. Calzarossa and E. Gelenbe, editors, *Performance Tools and Applications to Networked Systems: Revised Tutorial Lectures*, number 2965 in LNCS, pages 311–335. Springer-Verlag, 2004.

[41] S. Gilmore, J. Hillston, L. Kloul, and M. Ribaudo. Software performance modelling using PEPA nets. In *Proc. of 4th Int. Workshop on Software and Performance*, pages 13–24, Redwood Shores, California, USA, January 2004. ACM Press.

[42] S. Gilmore, J. Hillston, and M. Ribaudo. An efficient algorithm for aggregating PEPA models. *IEEE Transactions on Software Engineering*, 27(5):449–464, May 2001.

[43] S. Gilmore, J. Hillston, and M. Ribaudo. PEPA nets: A structured performance modelling formalism. In *Proc. of 12th Intl. Conf. on Modelling Tools and Techniques for Computer and Communication System Performance Evaluation*, number 2324 in LNCS, pages 111–130, London, UK, April 2002. Springer-Verlag.

[44] S. Gilmore, J. Hillston, M. Ribaudo, and L. Kloul. PEPA nets: A structured performance modelling formalism. *Performance Evaluation*, 54(2):79–104, October 2003.

[45] S. Gilmore and L. Kloul. A unified tool for performance modelling and prediction. In *Proc. of 22nd Intl. Conf. on Computer Safety, Reliability and Security (SAFECOMP 2003)*, volume 2788 of *LNCS*, pages 179–192. Springer-Verlag, 2003.

[46] S. Gilmore, L. Kloul, and D. Piazza. Modelling role-playing games using PEPA nets. In *Proceedings of the 19th International Symposium on Computer and Information Sciences (ISCIS 2004)*, volume 3280 of *LNCS*, pages 523–532, Kemer-Antalya, Turkey, October 2004. Springer-Verlag.

[47] N. Götz, U. Herzog, and M. Rettelbach. TIPP—a language for timed processes and performance evaluation. Technical Report 4/92, IMMD7, University of Erlangen-Nürnberg, Germany, November 1992.

[48] P. Harrison and J. Hillston. Exploiting Quasi-reversible Structures in Markovian Process Algebra Models. *The Computer Journal*, 38(6), 1995.

[49] P.G. Harrison. Turning back time in Markovian process algebra. *TCS*, 290:1947–1986, 2003.

[50] W. Henderson and P.G. Taylor. Embedded Processes in Stochastic Petri Nets. *IEEE Transactions on Software Engineering*, 17(2):108 – 116, February 1991.

[51] H. Hermanns, J. Meyer-Kayser, and M. Siegle. Multi-terminal binary decision diagrams to represent and analyse continuous time markov chains. In *Proc. of 3rd Intl. Workshop on the Numerical Solution of Markov Chains*, pages 188–207, 1999.

[52] U. Herzog. Formal description, time and performance analysis: A framework. Technical Report 15/90, IMMD VII, Friedrich-Alexander-Universität, Erlangen-Nürnberg, Germany, September 1990.

[53] J. Hillston. The nature of synchronisation. In U. Herzog and M. Rettelbach, editors, *Proc. of 2nd Int. Workshop on Process Algebras and Performance Modelling*, pages 51–70, Erlangen, 1994.

[54] J. Hillston. Compositional Markovian Modelling Using a Process Algebra. In W.J. Stewart, editor, *Numerical Solution of Markov Chains*. Kluwer, 1995.

[55] J. Hillston. *A Compositional Approach to Performance Modelling*. Cambridge University Press, 1996.

[56] J. Hillston. *FMPA Lecture Notes*, chapter Exploiting Structure in Solution: Decomposing Composed Models. Springer-Verlag, 2001.

[57] J. Hillston and L. Kloul. An efficient Kronecker representation for PEPA models. In de Alfaro and Gilmore [32], pages 120–135.

[58] J. Hillston, L. Kloul, and A. Mokhtari. Active nodes performance analysis using PEPA. In *Proc. of 19th UK Performance Engineering Workshop*, pages 244–256, 2003.

[59] J. Hillston, L. Kloul, and A. Mokhtari. Towards a feasible active networking scenario. *Telecommunication Systems*, 27(2–4):413–438, 2004.

[60] J. Hillston and V. Mertsiotakis. A simple time scale decomposition technique for stochastic process algebras. *The Computer Journal*, 38(7):566–577, 1995.

[61] J. Hillston and N. Thomas. A Syntactical Analysis of Reversible PEPA Models. In *Proc. of 6th Process Algebra and Performance Modelling Workshop*, Nice, France, September 1998. University of Verona.

[62] J. Hillston and N. Thomas. Product form solution for a class of PEPA models. *Performance Evaluation*, 35(3–4):171–192, 1999.

[63] C.A.R. Hoare. *Communicating Sequential Processes.* Prentice-Hall, 1985.

[64] D.R.W. Holton. A PEPA specification of an industrial production cell. *The Computer Journal*, 38(7):542–551, 1995.

[65] J. Hunter. Re-evaluation of the PEPA Workbench. Master's thesis, School of Computer Science, The University of Edinburgh, September 1999.

[66] J.R. Jackson. Jobshop-like Queueing Systems. *Management Science*, 10:131–142, 1963.

[67] S. Jarvis, editor. *Proc. of 19th UK Performance Engineering Workshop*, University of Warwick, July 2003.

[68] H. Jungnitz. *Approximation Methods for Stochastic Petri Nets.* PhD thesis, Rensselaer Polytechnic Institute, May 1992.

[69] F. Kelly. *Reversibility and Stochastic Processes.* Wiley, 1979.

[70] J.G. Kemeny and J.L. Snell. *Finite Markov Chains.* Van Nostrand, 1960.

[71] W.J. Knottenbelt. Generalised Markovian analysis of timed transition systems. Master's thesis, University of Cape Town, 1996.

[72] M. Kwiatkowska, G. Norman, and D. Parker. PRISM: Probabilistic symbolic model checker. In *Proc. of 12th Int. Conf. on Modelling Tools and Techniques for Computer and Communication System Performance Evaluation*, number 2324 in LNCS, pages 200–204, London, UK, April 2002. Springer-Verlag.

[73] K. Larsen and A. Skou. Bisimulation through probabilistic testing. *Information and Computation*, 94(1):1–28, 1991.

[74] V. Mertsiotakis. *Approximate Analysis Methods for Stochastic Process Algebras.* PhD thesis, Universität Erlangen–Nürnberg, Martensstraße 3, 91058 Erlangen, September 1998.

[75] V. Mertsiotakis and M. Silva. A Throughput Approximation Algorithm for Decision Free Processes. In M. Ribaudo, editor, *Proc. of 7th Int. Workshop on Petri Nets and Performance Models*, 1996.

[76] R. Milner. *Communication and Concurrency.* Prentice-Hall, 1989.

[77] B. Mitchell and J. Hillston. Analysing web service composition with PEPA. In *Proc. of 3rd Workshop on Process Algebras and Stochastically Timed Activities*, pages 33–44, Edinburgh, Scotland, June 2004.

[78] I. Mitrani and P.E. Wright. Routing in the Presence of Breakdowns. *Performance Evaluation*, 20:151–164, 1994.

[79] B.K. Oksendal. *Stochastic Differential Equations.* Springer, 2003.

[80] G. D. Plotkin. A Structural Approach to Operational Semantics. Technical Report DAIMI FN-19, University of Aarhus, 1981.

[81] M. Sereno. Towards a Product Form Solution of Stochastic Process Algebras. *The Computer Journal*, 38(6), 1995.

[82] T.T. Soong. *Random Differential Equations in Science and Engineering.* Academic Press, 1973.

[83] B. Strulo and P.G. Harrison. Spades - a process algebra for discrete event simulation. *J. Logic Computat*, 10(1):3–42, 2000.

[84] N. Thomas and S. Gilmore. Applying Quasi-Separability to Markovian Process Algebra. In *Proc. of 6th Process Algebra and Performance Modelling Workshop*, Nice, France, September 1998. University of Verona.

[85] N. Thomas and J. Hillston. Using Markovian process algebra to specify interactions in queueing systems. Technical Report ECS-LFCS-97-373, LFCS, The University of Edinburgh, 1997.

[86] F. Wan. Interface engineering and transient analysis for the PEPA Workbench. Master's thesis, School of Computer Science, The University of Edinburgh, September 2000.